Marburg University

Department of Computer Science
Prof. Dr. Sebastian Wild

Date: 2026-05-08
Version: 2026-05-08 22:46

Exercise Sheet 4 for Algorithmen und Datenstrukturen (Sommer 2026)

Hand In: *Until 2026-05-15 18:00, on ILIAS.*

Problem 1 (Stirling's bound)

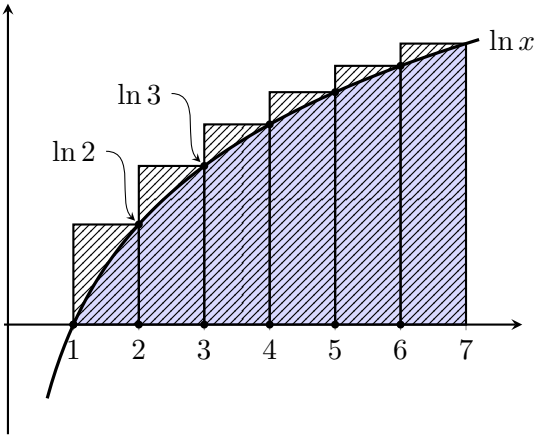
10 + 20 + 20 points

a) Prove: $\forall n \in \mathbb{N}_{\geq 0} : \ln(n!) \leq n \ln n$.

Hint: $\ln(xy) = \ln(x) + \ln(y)$

b) Prove: $\forall n \in \mathbb{N}_{\geq 2} : \ln(n!) \geq (n-1)\ln(n-1) - n$.

Hint: Use the fact that a sum $f(1) + \cdots + f(n)$ can be bounded from below by $\int_2^n f(x)dx$ for *concave* f.



c) Give a $\sim$ asymptotic approximation for $\ln(n!)$.

Hint: Use a) and b), as well as

$$\ln(n-1) \;=\; \ln\left(n \cdot \frac{n-1}{n}\right) \;=\; \ln(n) + \ln\left(1 - \frac{1}{n}\right) \;=\; \ln(n) \pm O\left(\frac{1}{n}\right)$$

(The last step follows form the Taylor series expansion of $\ln(x)$ around $x = 1$; for ADS, you do not necessarily need to know that 😉)

Problem 2 (Monty Hall) 20 points

You are a contestant on a game show. The host, Monty, shows you three closed doors.

Behind one door is a brand-new car.

Behind the other two doors are goats.

The game proceeds in 3 steps:

1. You pick a door (let's say Door #1). You hope the car is there, but you don't open it yet.
2. Monty, who knows exactly what is behind every door, opens one of the other doors (e.g., Door #3) to reveal a goat.
3. Monty then looks at you and asks: "Do you want to stick with Door #1, or would you like to switch to Door #2?"

Analyze your probability of winning the car for the two possible strategies: "Stay" and "Switch".

Problem 3 (Percolation) 50 points

Solve `marburg.kilonova.ro/problems/2` (Percolation).

ILIAS Submission

In your submission to ILIAS, describe your algorithm according to the template from class. The pseudocode part can here simply be an informal summary of your kilonova code submission.

Template for the design of algorithms

1. **Algorithmic Idea**
   Abstract idea that makes the algorithm work (prose)
   (an expert could fill in the rest from here)
2. **Pseudocode**
   structured description of procedure including edge cases
   should be unambiguous and close to real code
3. **Correctness proof**
   argument why the correct result is computed
   often uses induction and invariants
4. **Algorithm analysis**
   analysis of the efficiency of the algorithm
   usually want Θ-class of worst-case running time
   where interesting, also space usage